Ontology Building

Present by:
Umi Laili Yuhana
[1] Computer Science & Information Engineering National Taiwan University
[2] Teknik Informatika Institut Teknologi Sepuluh Nopember ITS Surabaya Indonesia
11/04/2007
Outline

• Reason
• Definition of Ontology
• Building Ontology
• Reference
Reason

- To Share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To Analyze domain knowledge
What is an Ontology

• Kind of things that actually exist, and how to describe them -> philosophy term

• In computer science:
  – Explicit and formal specification of a conceptualization
  – Consist of finite list of terms and the relationships between these terms
Building Ontology

Case study : Daycare Ontology
A small child care consisting:
- 3 classrooms (2 in the morning, 1 in the afternoon)
- 3 teachers (each teacher assign to 1 classroom)
- handfull of children
Some students have behavior that can endanger person. Each student has teacher.
Step by step Build Ontology

- Determine scope
- Consider reuse
- Enumerate Terms
- Define Taxonomy
- Define Properties
- Define Facets
- Define Instances
- Check for Anomalies
Step by step Build Ontology

- **Determine scope**
- Consider reuse
- Enumerate Terms
- Define Taxonomy
- Define Properties
- Define Facets
- Define Instances
- Check for Anomalies
Determine Scope (Q)

- Basic questions:
  - What is the domain that the ontology will cover?
  - For what we are going to use the ontology?
  - For what types question should the ontology provide answer?
  - Who will use and maintain the ontology?
Determine Scope (A)

• Answer:
  – What is the domain that the ontology will cover? → *Small child care center / daycare*
  – For what we are going to use the ontology?
    • Infer knowledge about student’s negative behaviors to which s/he will be exposed
  – For what types question should the ontology provide answer?
    • Who is the classmates of each student?
    • What is negative behavior of each student?
  – Who will use and maintain the ontology?
    • Teachers, to care the student and avoid the student from negative behavior of her/his classmate
Step by step Build Ontology

- Determine scope
- **Consider reuse**
- Enumerate Terms
- Define Taxonomy
- Define Properties
- Define Facets
- Define Instances
- Check for Anomalies
Consider Reuse

• We can reuse ontology in the same domain knowledge if it exist
• If no ontology exist, create new one
Step by step Build Ontology

- Determine scope
- Consider reuse
- **Enumerate Terms**
- Define Taxonomy
- Define Properties
- Define Facets
- Define Instances
- Check for Anomalies
Enumerate Terms

• Identify relevant terms
• Write down in an unstructured list all the relevant terms
• Noun -> basis for class names
• Verbs -> basis for property names
  – Is part of
  – Has component
  – etc
A small child care consisting:
- 3 classrooms (2 in the morning, 1 in the afternoon)
- 3 teachers (each teacher assign to 1 classroom)
- handful of children
Some students have behavior that can endanger person. Each student has teacher.
Enumerate Term (cont.)

- Classroom
- Student
- Teacher
- Behavior
- Person
- behavior_of
- has_behavior
- is_practice_by
- endanger

- is_exposed_to
- teach
- is_taught
- includes_student
- attends_classroom
- has_teacher
- teach_student
- has_classmate
- has_date_of_birth
- has_age
Step by step Build Ontology

- Determine scope
- Consider reuse
- Enumerate Terms
- **Define Taxonomy**
- Define Properties
- Define Facets
- Define Instances
- Check for Anomalies
Define Taxonomy

- Organize relevance terms in taxonomic (subclass) hierarchy
- Terms as class: Classroom, Student, Teacher, Person, Behavior
Define Taxonomy (cont.)
Step by step Build Ontology

• Determine scope
• Consider reuse
• Enumerate Terms
• Define Taxonomy
• Define Properties
• Define Facets
• Define Instances
• Check for Anomalies
Property

- Property: represent relationships between two individual
- Property = Slot
- Link individual from the domain and individual from the range
- There are 3 properties:
  - Object Properties
  - Data Type Properties
  - Annotation Properties
Object Property

- Link an individual to an individual
- Types of object property
  - Inverse property
    - e.g. : has_parent inverse of has_child
  - Functional property
    - e.g. : has_birth_mother
  - Transitive property
    - e.g. : has_anchestor
  - Symmetric property
    - e.g. : has_sibling
Data Type Property

• Link an individual to an XML schema data type value or an rdf literal
• e. g. :

```
has_birth_of_date
```

```
Student
```

```
date
```
Annotation Property

• Used to add information (metadata – data about data) to classes, individuals and object / data type property
<table>
<thead>
<tr>
<th>No</th>
<th>Property</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>has_behavior</td>
<td>Student</td>
<td>Behavior</td>
</tr>
<tr>
<td>2</td>
<td>is_practice_by</td>
<td>Behavior</td>
<td>Student</td>
</tr>
<tr>
<td>3</td>
<td>endanger</td>
<td>Behavior</td>
<td>Person</td>
</tr>
<tr>
<td>4</td>
<td>is_exposed_to</td>
<td>Person</td>
<td>Behavior</td>
</tr>
<tr>
<td>5</td>
<td>teach</td>
<td>Teacher</td>
<td>Classroom</td>
</tr>
<tr>
<td>6</td>
<td>is_taught</td>
<td>Classroom</td>
<td>Teacher</td>
</tr>
<tr>
<td>7</td>
<td>Includes_student</td>
<td>Classroom</td>
<td>Student</td>
</tr>
<tr>
<td>8</td>
<td>attends_classroom</td>
<td>Student</td>
<td>Classroom</td>
</tr>
<tr>
<td>9</td>
<td>has_teacher</td>
<td>Student</td>
<td>Teacher</td>
</tr>
<tr>
<td>10</td>
<td>teach_student</td>
<td>Teacher</td>
<td>Student</td>
</tr>
<tr>
<td>11</td>
<td>has_classmate</td>
<td>Student</td>
<td>Student</td>
</tr>
<tr>
<td>12</td>
<td>has_date_of_birth</td>
<td>Student</td>
<td>Date</td>
</tr>
<tr>
<td>13</td>
<td>has_age</td>
<td>Student</td>
<td>Int</td>
</tr>
</tbody>
</table>
Define Property (cont.)

- **has_behavior** is inverse of **is_practice_by**
- **endanger** is inverse of **isExposed_to**
- **teach** is inverse of **is_taught**
- **includes_student** is inverse of **attends_classroom**
- **has_teacher** is inverse of **teach_student**
- **has_classmate** is symmetric property
- **has_date_of_birth** is data type property
- **has_age** is data type property
Object Properties

- attends_classroom <-> include_student
- include_student <-> attends_classroom
- is_practice_by <-> has_behavior
- has_behavior <-> is_practice_by
- is_exposed_to <-> endanger
- endanger <-> is_exposed_to
- teach_student <-> has_teacher
- has_teacher <-> teach_student
- is_taugh_by <-> teach_classroom
- teach_classroom <-> is_taugh_by
- has_classmate <-> has_classmate
Data Type Property

Ontology Building - Yuhana
Step by step Build Ontology

• Determine scope
• Consider reuse
• Enumerate Terms
• Define Taxonomy
• Define Properties
• **Define Facets**
• Define Instances
• Check for Anomalies
Facets

• Facet is used to represent information about properties (slots), sometimes called role restrictions

• Kind of facets:
  – Cardinality
  – Value Type
Cardinality

- Cardinality represents the **exact number of values** that may be **asserted** for the **slot** for that class
  - Single cardinality
  - Multiple cardinality
    - Minimum cardinality
    - Maximum cardinality
Value Type

• Value-type facet describes what type of values can fill in the slot
  – String
  – Number
  – Boolean
  – Enumerated
Cardinality

• Cardinality represents the **exact number of values** that may be **asserted** for the **slot** for that class

• Cardinality
  • Minimum cardinality
  • Maximum cardinality

• Value Type
Step by step Build Ontology

- Determine scope
- Consider reuse
- Enumerate Terms
- Define Taxonomy
- Define Properties
- Define Facets
- **Define Instances**
- Check for Anomalies
Instance

• Object or individual of class
• Example:
  – Instances of student:
    • Ariel,
      – has_date_of_birth : November 15, 2004
      – has_behavior : throwing_toys
      – Has_teacher : miss_Lyn
# Instances

<table>
<thead>
<tr>
<th>Instance of Class</th>
<th>is_taught_by</th>
</tr>
</thead>
<tbody>
<tr>
<td>combine_PM_classroom</td>
<td>miss_Julie</td>
</tr>
<tr>
<td>older_kids_AM_classroom</td>
<td>miss_Mandy</td>
</tr>
<tr>
<td>younger_kids_AM_classroom</td>
<td>miss_Lyn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance of Student</th>
<th>has_date_of_birth</th>
<th>has_behavior</th>
<th>has_teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>ariel</td>
<td>November 15, 2004</td>
<td>Throwing_toys</td>
<td>miss_Lyn</td>
</tr>
<tr>
<td>cal</td>
<td>September 12, 2003</td>
<td></td>
<td>miss_Mandy</td>
</tr>
<tr>
<td>cass</td>
<td>January 20, 2005</td>
<td>Throwing_toys</td>
<td>miss_Julie</td>
</tr>
<tr>
<td>ella</td>
<td>June 15, 2004</td>
<td>Bitting, Pinching</td>
<td>miss_Lyn</td>
</tr>
<tr>
<td>ginny</td>
<td>December 20, 2003</td>
<td>Hitting</td>
<td>miss_Mandy</td>
</tr>
<tr>
<td>jeremy</td>
<td>April 24, 2003</td>
<td>Throwing_toys</td>
<td>miss_Mandy</td>
</tr>
<tr>
<td>katie</td>
<td>March 14, 2005</td>
<td>Bitting</td>
<td>miss_Lyn</td>
</tr>
<tr>
<td>nate</td>
<td>December 22, 2003</td>
<td></td>
<td>miss_Mandy, miss_Julie</td>
</tr>
<tr>
<td>scott</td>
<td>September 9, 2004</td>
<td>Bitting</td>
<td>miss_Lyn, miss_July</td>
</tr>
<tr>
<td>zach</td>
<td>July 10, 2003</td>
<td></td>
<td>miss_Julie, miss_Mandy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance of Teacher</th>
<th>teaches_classroom</th>
<th>teach_students</th>
</tr>
</thead>
<tbody>
<tr>
<td>miss_Julie</td>
<td>combine_PM_classroom</td>
<td>cass, nate, scott, zach</td>
</tr>
<tr>
<td>miss_Lyn</td>
<td>younger_kids_AM_classroom</td>
<td>ariel, ella, katie, scott</td>
</tr>
<tr>
<td>miss_Mandy</td>
<td>older_kids_AM_classroom</td>
<td>cal, ginny, jeremy, nate, zach</td>
</tr>
</tbody>
</table>
Fill instances in protégé
Step by step Build Ontology

• Determine scope
• Consider reuse
• Enumerate Terms
• Define Taxonomy
• Define Properties
• Define Facets
• Define Instances
• Check for Anomalies
Check Anomalies

• Check anomalies or consistency with reasoner
  – Pellet
  – Racer
Conclusion

• There is no single correct ontology for any domain
• Quality of ontology can be proofed by using it in applications
Reference

• Horridge, Mattahew, A Practical Guide to Building OWL Ontologies Using The Protege-OWL plugin and CO-ODE Tool, The University of Manchester, 2004
• https://mywebspace.wisc.edu/jpthielman/web/DaycareOntology.htm